SECURING EMBEDDED LINUX

Mike Anderson
Chief Scientist

The PTR Group, Inc.
http://ThePTRGroup.com
mike@theptrgroup.com

Who is The PTR Group?

- The PTR Group was founded in 2000
- We are involved in multiple areas of work:
  - Robotics (NASA space arm)
  - Flight software (over 35 satellites on orbit)
  - Defensive cyber operations
    - I’ll leave this to your imagination ☺
  - Embedded software ports to RTOS/Linux/bare metal
  - IoT systems architecture and deployment

Who am I?

- Over 39 years in the embedded space
- Long-time developer in the RTOS field
- Instructor for Linux/Android internals
- Mentor for FRC #116 FIRST Robotics Team
- Frequent speaker at:
  - Embedded Linux Conference
  - Embedded Systems Conference
  - CIA Emerging Technology Conference
  - And more…

What We'll Talk About…

- What does it mean to be secure?
- What are we trying to protect?
- Who are the attackers?
- Physical access
- Secure boot techniques
- Encryption, certificates, code signing and digital signatures
- Characteristics of a secure system
- Steps to secure the data center, border gateway and the edge devices

The Dimensions of Security

Source: newgrounds.com

- The definition of security varies depending on the audience
- For some, it means having locks, alarms and guards as in physical security
- For others, it is all about protection from outside hackers as in cyber security
- Many will confuse privacy with security
  - They're related, but not the same thing
- There is a spectrum for security
  - Usability on one end and protection on the other

Security Facets

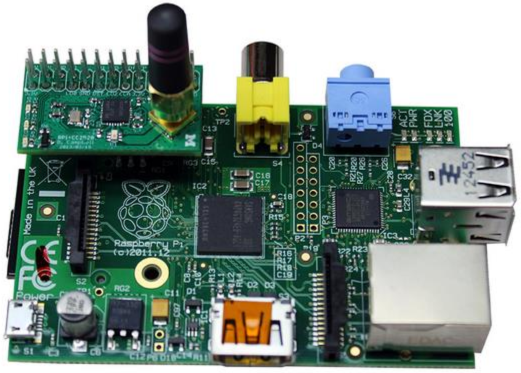
- If we think about security's many dimensions, we need to consider the following elements:
  - Confidentiality
  - Integrity
  - Authentication
  - Authorization
  - Non-repudiation
- Each of these topics need to be addressed at some level to be able to assert that a system is "secure"

Embedded Linux Devices

- Naturally, Linux can be found in thousands of devices
  - It became the feature-rich embedded OS of choice a few years ago
- But, how do we define "embedded"?
  - Essentially, if you inherently know there's a computer in there someplace, but don't see a keyboard, mouse, and monitor, it's probably embedded
  - This draws into question about the status of smartphones and tablets, but let's not go there

Example: IoT Border Routers

- The gateways between the edge and the Internet (cloud)
  - Take in low-power wireless on one side and spit out IP via Wi-Fi or Ethernet
- Due to non-IP routable protocols like ZigBee and Z-Wave, edge devices can't get to the cloud directly
  - They need a translation service to collect, collate and retransmit the data using IPv4/IPv6
  - The border gateways are key in the "fog" model
- The border routers can also provide for command and control of the edge devices
  - Like the Nest thermostat

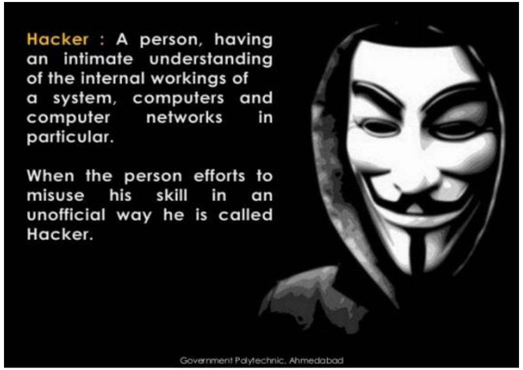


Source: bradcampbell.com

Who are the Attackers?

- Amateur hackers
  - Most of the hacking that done on the Internet currently is the effort of amateur hackers
  - A.k.a., “Script Kiddies”
- Professional hackers
  - Blackhats
    - Ransomware
    - Credit card thieves
    - Malware-to-order
  - Whitehats (theoretically, not a bad guy)
    - “Ethical” hackers
    - Frequently employed to detect security vulnerabilities
    - Pentesters
  - Grayhats
    - Living in the gray area between the legal and illegal
- State-sponsored hackers
  - Often blackhats that are paid by Governments to find and obtain classified information, conduct industrial espionage or launch coordinated cyber-attacks
    - Not in it for the money per se

Source: freethoughtblogs.com



Source: slideshare.net

Understanding what “Security” Means

- The ecosystem embedded devices present unique challenges for security
  - Some challenges are easy to understand, but others are more subtle
- However, there are some tenets that apply across the board
- How you address these common elements varies widely from organization to organization

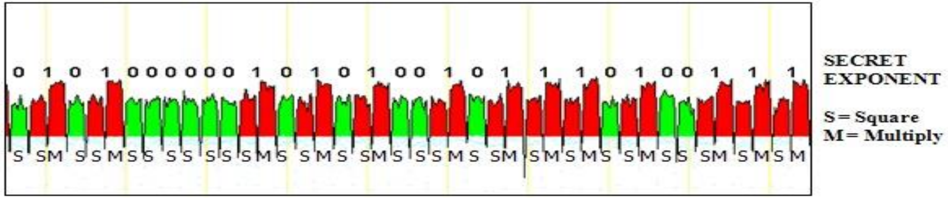
Insider Threat

- Something that every organization must come to grips with is the “insider threat”
  - Yes, your own employees
    - They know the signature algorithms
    - Have access to the signing keys
    - They know where the debug interfaces are
    - Can put backdoors like door-knock protocols into the code
- Peer review is one of the easiest ways to mitigate some of these issues
  - Also helps identify potential coding errors that could lead to vulnerabilities
  - This makes a great case for the use of open-source projects
- Two-person rules for accessing keys also help limit the insider threat

Source: phys.org

Physical Access is a Problem

- Any time you allow physical access to a sensor, data processing or network communications equipment you open up security vulnerabilities
- There are a number of physical attacks against computer platforms that simply can't be done remotely
  - Ranging from simply unplugging power to sophisticated electromagnetic techniques such as Differential Power Analysis

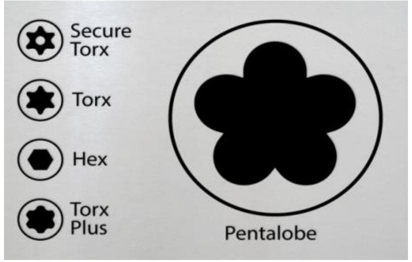


Source: eetimes.com

Physical Access #2

- Techniques to thwart physical access include:
  - Encasing the device in epoxy
    - Also called "potting" the device
  - Adding anti-tamper sensors
  - Placing the device in an anti-tamper case
  - Using special screws
  - Special adhesives
  - Removing debugging interfaces
  - Blowing the e-fuses
- All of these can be defeated given enough time
  - Assume that your device will be compromised sooner or later

Source: teacoinc.com



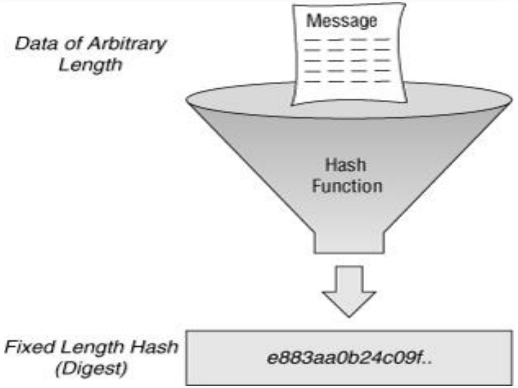
Source: bobmackay.com

Secure Boot Techniques

- There is typically a window of vulnerability for any system during the boot sequence
  - Fortunately, there are now techniques to address this
- There are several approaches to ensuring that the computer boots with known-good software images
  - Most of these rely on the availability of security hardware such as a smart card or Trusted Platform Module (TPM)

Secure Boot Techniques #2

- These will typically compute a cryptographic hash of a piece of firmware/software to “measure” it
  - They then compare hash results to a value located in secure data storage
    - Cryptographically protected or written to One-Time Programmable (OTP) memory



Source: cisco.com

- If the hash matches the value from the secure storage, then the boot proceeds
  - If not, then an alert is signaled and the boot terminates
    - May use techniques like Intel’s AMT to signal System Admin via network

Secure Boot Techniques #3

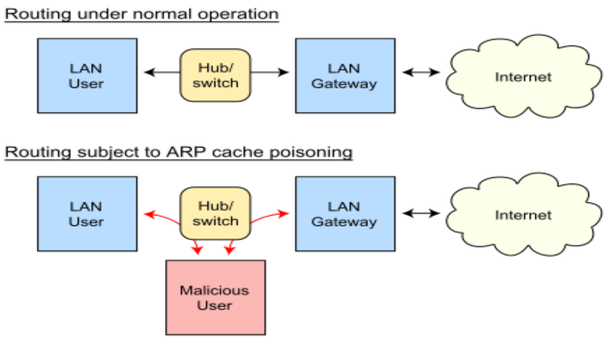
- However, normal operating system or boot firmware updates are greatly complicated because of the need to update the secure store
  - Authentication issues typically requiring a digital signature and/or a physical token
- There are also other potential issues
  - Microsoft's Trusted Boot that would only boot Microsoft's OS because it required Microsoft's digital signature
  - The "Tivo" effect that caused GPLv3 creation

Confidentiality

- This is probably one of the easiest characteristics of security to understand
- The goal of confidentiality is simply that no unauthorized individuals can read the data you want protected
- This data breaks down to:
  - Data-in-flight
  - Data-at-rest
- Encryption technology is most closely related to issues of confidentiality
- Confidentiality is often associated with privacy
  - But, we can achieve privacy without encryption

Confidentiality – Data-in-Flight

- This refers to protecting message traffic from being read as it is sent/received
- This is where the man-in-the-middle (MITM) attacks are most common
  - DNS spoofing, ARP spoofing, packet interception, etc.
- Encrypting the links is the most common approach to data-in-flight confidentiality
  - However, public Internet routers will not be able to decrypt your packets, so they must be wrapped inside of a readable packet
- VPNs are a typical implementation
- Need to be aware of OPSEC issues



Source: wikipedia.com

Confidentiality – Data-at-Rest

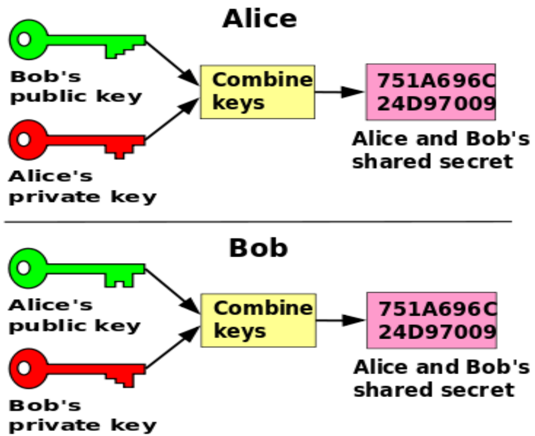
- Data-at-rest refers to sensitive data that is actually stored on the device rather than simply transiting through the device
  - Encryption keys, passwords, collected data, etc.
- Any sensitive data-at-rest should be encrypted
  - Improves confidentiality even if physical access is allowed
- You can encrypt the entire data storage device, specific directories or specific files
  - E.g., Linux eCryptFS or PGP

Encryption Background

- Without getting too far into the topic, encryption can be thought of as either symmetric or asymmetric
- With symmetric encryption, there is a pre-shared key that must be known on both sides
  - E.g., AES, DES, Twofish, Rijndael, Triple-DES
  - The question is how to exchange the keys?
- In asymmetric encryption, we have public and private keys often referred to as public-key cryptography
  - E.g., Diffie-Hellman, RSA, elliptic curve

Public-Key Cryptography

- The idea behind public-key cryptography is that you have a public and a private key
- Anyone can encrypt a message using your public key
  - But only you can decrypt the message using your private key
- A modification to this is the Diffie-Hellman key exchange approach
  - To send to a recipient, you encrypt with your private key and their public key
  - They then decrypt with your public key and their private key
  - Allows you to set up an encrypted session for the exchange of symmetric keys



Source: wikipedia.com

Public-Key Encryption #2

- Neither you nor the recipient need to share your private keys
- Public keys can be stored on a public server
- Has provisions for the key to expire as well as being able to revoke the keys if they're compromised
- The Diffie–Hellman key exchange approach provides for non–repudiation as well
  - Only you have your private key and therefore, only you could have generated the message

Trouble in Paradise…

- Many asymmetric approaches are predicated on the difficulty of factoring large prime numbers
  - Or, some other computationally difficult problem like elliptical curve computation
- However, the rise of quantum computing is threatening the viability of asymmetric crypto
  - Shor’s Algorithm can factor large primes
  - This means that someday soon they could be able to break your Diffie-Hellman key exchange in real time
- Currently, quantum computers are still small
  - But, in 5-10 years, quantum computers will be sufficiently capable as to be able to break RSA and similar algorithms

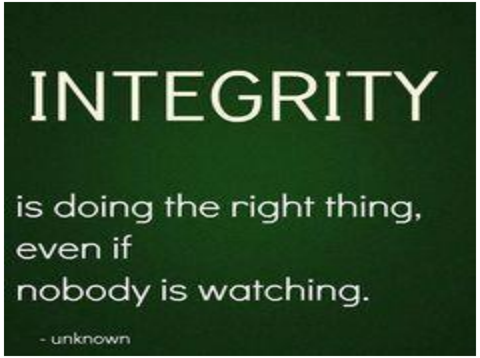
Source: extremetech.com

Services Provided by Encryption

- Link encryption (data–in–flight) certainly does provide for confidentiality
  - However, there are some additional benefits
- If we can guarantee that the key has not been compromised, then we also gain a measure of authentication
  - Only a device with the secret key could have generated a message that decrypts with the same key
- Additionally, we also gain some message integrity checking
  - If the message was modified in flight, it will not decrypt properly

Integrity

- Integrity encompasses a couple of different concepts
  - System integrity
  - Message integrity
- System integrity can be addressed initially by ensuring a secure boot cycle
  - More on this is coming up
- Message integrity is a somewhat different matter
  - We need to concern ourselves that the message was delivered intact
  - And, we need to ensure that the message wasn't modified



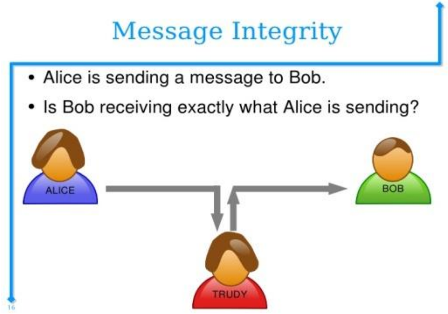
Source: pinterest.com

System Integrity

- In order for the user to associate some level of trust in the system's integrity, we should expect that the computer system:
  - Has a vetted boot cycle with steps to ensure the boot firmware and OS are unmodified
    - A secure file system is also a plus
  - Physical access is restricted to the greatest extent possible
    - Tamper-proof case, etc.
  - That power is reasonably reliable
  - Network connectivity isn't easily compromised
  - The system has a means to authenticate users and commands

Message Integrity

- Making sure that the message is intact can be addressed through the many checksums or CRCs that we find associated with typical messaging techniques
  - Ethernet CRC, IP header checksum, UDP payload checksum
- However, making sure the message wasn't modified in-flight by a MITM is more complicated
  - As outlined earlier, one approach is to encrypt the message
  - Another is to associate a message integrity code (MIC) or other hash-based message authentication code (HMAC) with the message



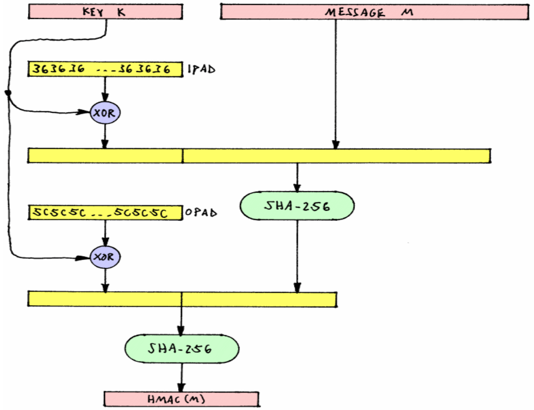
Source: slideshare.com

MICs, MACs and HMACs

- The term Message Integrity Code (MIC) is often used interchangeably with the term message authentication code (MAC)
  - MIC is often used by networking people to avoid confusion with a Media Access Controller
- MICs are typically based on using a symmetric key to calculate a keyed hash of the message
  - Only the receiver with the secret key can recalculate the hash to determine if the message was modified
    - Therefore, they are unforgeable
  - These are not to be confused with digital signatures that use asymmetric encryption techniques

MICs, MACs and HMACs #2

- HMACS are irreversible hash-based MACs that calculate a value of the message as a message digest
  - You cannot determine the message by looking at the hash
  - The message digest is typically not included with the message, but must be retrieved separately and compared
    - E.g., SHA-1, MD5, SHA-256, etc.



Source: cs/rit.edu

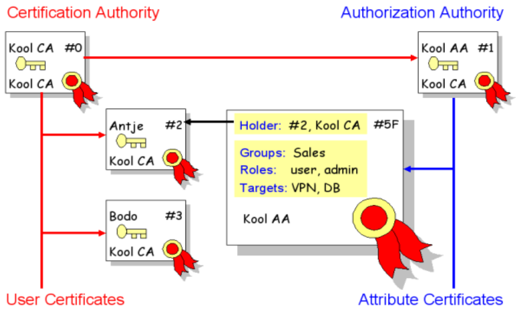
Authentication

- Authentication addresses being able to associate a message, user or file's origin to a valid source
- Single-factor authentication uses a single characteristic to grant access
  - Password, biometrics, PINs, security tokens, etc.
- However, passwords are generally regarded as too weak for security usage
  - They can easily be compromised due to poor passwords or user's indiscretion
- Two-factor authentication combines two characteristics to provide more security than a password alone
  - E.g., Fingerprint and a drawn pattern or iris scan and a security token such as a smart card

Source: halfelf.org

Authentication #2

- While we can get some level of authentication through the use of encryption, the typical mechanism used in system authentication is the certificate



Source: free-it.org

- Certificates are closely tied to the asymmetric encryption approach
  - Certificates are a way of binding a particular public key to a specific distinguished name or its alternative such as an e-mail address or DNS entry

X.509 Certificates

- ITU-T X.509 was originally issued in 1988 and is associated with the X.500 standard
  - Covered in IETF RFC 5280
- A certification authority (CA) issues a certificate binding a public key to a specific entity
  - The entity helps generate the certificate via their private key and this becomes the entity’s root certificate
- Certificates that are signed by the root certificate then establish a chain of trust
  - This creates a tree structure where the root certificate is at the top of the tree
- The issuer also has the ability to revoke a certificate that may have been compromised

Digital Signatures and Code Signing

- Digital signatures are a way of implementing an electronic signature
  - Legally binding in many countries
- Using asymmetric crypto, digital signatures are based on the same public/private key approach as certificates
  - As long as the private key hasn't been compromised, you also get non-repudiation
  - Can also use X.509 certificates
- This approach can also be extended to the concept of code signing
  - Provides for trusted identification based on the keys associated to a CA or other public cryptographic key
  - Used heavily in Linux repositories to verify authenticity of the code
  - Also used in .NET and is verified on first run of software

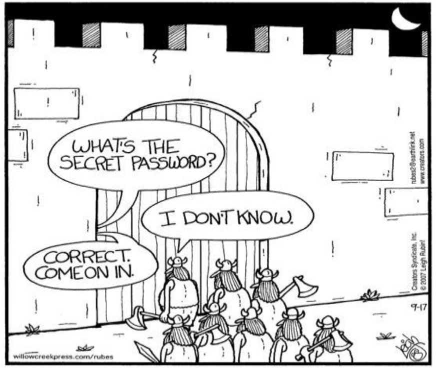
Source: chmag.in

Authorization

- Authorization is somewhat more difficult to pin down
  - Essentially, authorization is related to the system's access control policies
- It's assumed that a user/system is authorized if they have:
  - Valid credentials for the platform or
  - The appropriate secret key or
  - Knowledge of some private characteristic of the system such as a hidden SSID in a wireless network
    - Typically tied with a form of authentication such as a pre-shared key

Non-Repudiation

- Recall that non-repudiation means someone not being able to deny that they sent the message
- At this point, we've already seen several mechanisms that provide for non-repudiation
  - Often associated with having the private secret needed to encrypt/decrypt a cryptographically sealed message
- If we use two-factor authentication to further limit the range of possibilities, we can increase the level of non-repudiation
  - E.g., biometrics and the secret key



Why great care and consideration should be taken when selecting the proper password

Source: wordpress.com

Security is all about Risk Management

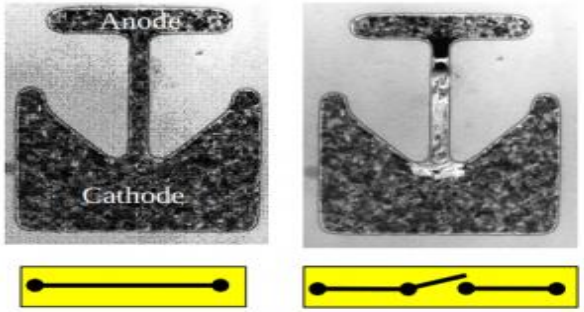
- Let's face it, security can be expensive
  - But, the lack of security can also be expensive
  - Specialized encryption equipment, additional software development, frequent software updates, continuing testing and monitoring of hacking sites, etc.
- It's quite easy for the costs of security measures to outweigh other development costs
- However, the amount of money that you spend is related to your company's internal risk-management beliefs
  - You have to decide what's important based on the potential amount of damage if the system is compromised
    - E.g., US OPM and Ashley-Madison

Implementing Linux Security

- Depending on their implementation, Linux platforms have characteristics in common regardless of their use
  - Linux is Linux
- GNU/Linux systems can have a lot of applications installed
  - A large attack surface
- Typically powered on all the time
  - But, may also be powered-off for extended periods of time
  - Power/thermal management policies need to be considered
- Assume that there will be physical access by non-authorized personnel
- Plan for spares so compromised devices can be replaced and sent back to HQ for analysis and remediation

Physical Security for Embedded Devices

- Unlike Linux-based data centers, there may not be any enforceable physical boundaries
- Remove any debugging interfaces
  - Blow the e-fuses, if available, to prevent access to internal registers or storage
- Place the unit in a tamper-resistant case
  - Use potting, special screws, etc.
- Assume the device will be compromised physically



Source: unlockfiretv.com

Data Security on Embedded Devices

- Implement a secure boot mechanism
- Use virtualization or containers to isolate communities of interest
  - The use of the "micro-server" concept
- Eliminate all non-essential services and software
- Periodic auditing of installed software
- Monitor and install software updates for the system regularly
- Two-factor authentication for accessing the system
- Implement mandatory access controls and auditing
  - SELinux or similar MAC system with access control lists and regular review of audit logs

Data Security on Embedded Devices

- Depending on the CPU horsepower and amount of data-at-rest on the device, you should encrypt the data store
  - At the partition, directory or file level as appropriate
  - Especially for store-and-forward applications
- Implement certificates for authentication
  - Make sure you have a solid certificate revocation approach to deal with compromised devices
- Implement code signing for updates
  - Keep gateways up to date with security patches
- Use mandatory access control and ACL policies, if possible

Network Security for Embedded

- Know what devices are on your network
  - Periodically re-inventory to detect new devices
- Implement IPv4 and IPv6 firewall policies
- Install intrusion detection/prevention system
  - E.g., snort
- Plan for periodic updates to your networking equipment firmware
- Close all non-essential ports and network services
  - Scan devices with tools like nmap, SATAN, SAINT, etc.
- Use VPNs for extended-term communications link requirements
- Use DTLS/TLS/AES for temporary-link security
- Use certificates for verification of the other end of the pipe
- Consider hiring penetration testers periodically

Network Security for Embedded #2

- If CPU and storage permit, implement IDS/IPS
  - Periodic virus/malware scans as well
- Monitor the network and track devices as they check in
  - Report new devices found on the network immediately
  - Track IDs of compromised devices and inform the rest of the network of the compromise
- Use encryption/DTLS/TLS for all links to the edge devices
- Use a VPN or encryption/TLS to talk to the data center
- Implement log management to prevent potential DoS attacks filling up the logs and crashing the gateway
- Use non-routable network on the edge links

Summary

- By now, you should be aware that securing the IoT and the various parts can be a daunting task
  - Security will cost you money, but your risk assessment will determine the risks you are willing to live with in your system
- Use a "fog" model to limit the attack surfaces of your device network
  - All direct communications are from the gateway or other edge nodes
    - Do not allow your edge devices to be visible on the Internet
- Understand how certificates work and develop a strategy for updating devices in the field
- Ask for and expect to pay for help in developing a security strategy
- Periodically, hire pentesters to verify your system against the latest techniques